online softmax from math to code

wenqingqian May 2024
code see Github

Contents

1 Math: from 3 pass safe softmax to 2 pass online softmax

3 pass safe softmax

for i in $[1, N]$:

$$m_i = \max(x_i, m_{i-1}) \tag{1}$$

for i in $[1, N]$:

$$d_i = d_{i-1} + e^{x_i - m_N} \tag{2}$$

for i in $[1, N]$:

$$y_i = \frac{e^{x_i - m_N}}{d_N} \tag{3}$$

The equation 2 equals to:

$$d_N = \sum_{i=1}^{N} e^{x_i - m_N} \tag{4}$$

which comes from:

$$d_j = \sum_{i=1}^{j} e^{x_i - m_N} \tag{5}$$

Actually we can create another squence to remove the dependency on N:

$$d'_j = \sum_{i=1}^{j} e^{x_i - m_j} \tag{6}$$

when $j = N$, $d_j = d'_j$.

Then we can find the recurrence relation between d'_j and d'_{j-1}:

$$\begin{aligned}
d'_j &= \sum_{i=1}^{j} e^{x_i - m_j} \\
&= \sum_{i=1}^{j-1} e^{x_i - m_j} + e^{x_j - m_j} \\
&= \sum_{i=1}^{j-1} e^{x_i - m_{j-1} + m_{j-1} - m_j} + e^{x_j - m_j} \\
&= \sum_{i=1}^{j-1} \left(e^{x_i - m_{j-1}} * e^{m_{j-1} - m_j} \right) + e^{x_j - m_j} \\
&= \left(\sum_{i=1}^{j-1} e^{x_i - m_{j-1}} \right) * e^{m_{j-1} - m_j} + e^{x_j - m_j} \\
&= d'_{j-1} * e^{m_{j-1} - m_j} + e^{x_j - m_j}
\end{aligned} \tag{7}$$

2 pass online softmax

for i in $[1, N]$:

$$\begin{aligned}
m_i &= \max(x_i, m_{i-1}) \\
d'_i &= d'_{i-1} * e^{m_{i-1} - m_i} + e^{x_i - m_i}
\end{aligned} \tag{8}$$

for i in $[1, N]$:

$$y_i = \frac{e^{x_i - m_N}}{d'_N} \tag{9}$$

2 Code implementation

This algorithm is easy to implement in CPU, but in GPU, to make the algorithm parallel, we need to generalize the original algorithm and use some CUDA tools.

In equation 8, each time we process a new number x_i, and calculate the new max number of sequence and summation. To make the algorithm parallel, we need to get the new result by two sequence directly, assuming there are two sequences:

- S with
  length: l_S
  max: S.max
  summation: $S.\text{sum} = \sum_{i=1}^{l_S} e^{s_i - S.\max}$

- B with
  length: l_B
  max: B.max
  summation: $B.\text{sum} = \sum_{i=1}^{l_B} e^{b_i - B.\max}$

The sequence $A = S + B$, assuming S.max is smaller than B.max.

- A
  length: $l_S + l_B$
  max: $A.\max = B.\max$
  summation: $A.\text{sum} = \sum_{i=1}^{l_S + l_B} e^{a_i - A.\max}$

Then the summation of A equals to:

$$
\begin{aligned}
A.\text{sum} &= \sum_{i=1}^{l_S+l_B} e^{a_i - A.\max} \\
&= \sum_{i=1}^{l_S+l_B} e^{a_i - B.\max} \\
&= \sum_{i=1}^{l_S} e^{s_i - B.\max} + \sum_{i=1}^{l_B} e^{b_i - B.\max} \\
&= \sum_{i=1}^{l_S} e^{s_i - S.\max + S.\max - B.\max} + \sum_{i=1}^{l_B} e^{b_i - B.\max} \\
&= \left(\sum_{i=1}^{l_S} e^{s_i - S.\max} \right) * e^{S.\max - B.\max} + \sum_{i=1}^{l_B} e^{b_i - B.\max} \\
&= S.\text{sum} \cdot e^{S.\max - B.\max} + B.\text{sum}
\end{aligned} \tag{10}
$$

Generally, we do not assume that $B.\max$ is greater than $S.\max$:

$$
\begin{aligned}
A.\max &= \max(B.\max, S.\max) \\
A.\text{sum} &= S.\text{sum} \cdot e^{S.\max - A.\max} + B.\text{sum} \cdot e^{B.\max - A.\max}
\end{aligned} \tag{11}
$$

Then we can process the sequence by multiple threads through warp_reduce.

online softmax CUDA pseudo

Assuming *input* sequence length $N = 32$

In each thread $i \in [1, 32]$
    local max = $input[i]$
    local sum = $e^0 = 1$
for (mask = $32 \gg 1$; mask ≥ 1; mask $\gg = 1$) **do**
  sum_tmp = __shfl_xor_sync(0xffffffff, local sum, mask)
  max_tmp = __shfl_xor_sync(0xffffffff, local max, mask)
  newmax = max(local max, max_tmp)
  local sum = local sum$\cdot e^{\text{local max - newmax}}$ + sum_tmp$\cdot e^{\text{max_tmp - newmax}}$
  local max = newmax

10: **end for**
11: In each thread $i \in [1, 32]$
12: $output[i] = (e^{input[i] - \text{local max}})/(\text{local sum})$

3 Profiling

As shown in the fig1, when the sequence size is larger than 512*16, the performance of online softmax is worse than 3 pass softmax. Since it is difficult to synchronize sum and max information between blocks, our algorithm only uses one block and 128 threads. When the data increases, multiple data will be allocated to one thread, and the processing of these data is serial, causing the time consumption to increase exponentially.

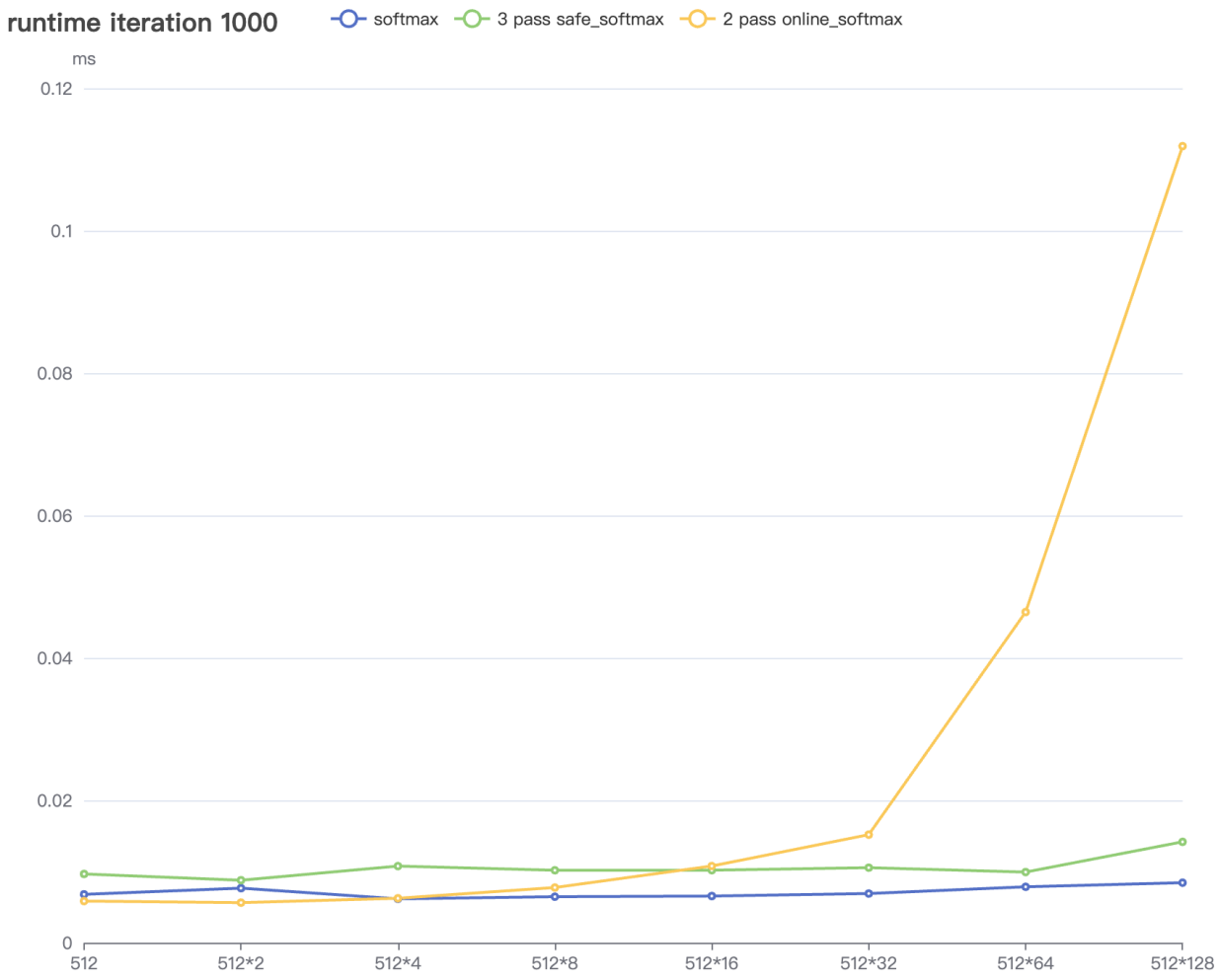

Figure 1: Profiling